

Computer Science Honours Report:
Assisting World Wide Web Navigation

Carey Evans

November 1, 1996

Abstract

The rapid growth of the World Wide Web to the large collection of loosely connected pages of information it is today has accentuated the problem of getting “lost in hyperspace.” Popular browsers do not provide the support needed to alleviate this problem.

This report reviews work on this problem, and presents a possible solution in a program “RedBack” which was developed to investigate methods to support WWW browsing. The design and implementation of the program and issues arising from its development are described.

Contents

1	Introduction	1
2	Previous work	3
2.1	Hypertext and World Wide Web	3
2.2	Navigation	3
2.3	Visualisation	5
3	RedBack design	7
3.1	Program scope	7
3.2	Code organisation	7
3.3	Graphical display	8
4	RedBack implementation	9
4.1	Implementation environment	9
4.1.1	NCSA Mosaic	9
4.1.2	Perl	11
4.1.3	libwww-perl	12
4.2	The RedBack program	13
4.2.1	CCI::Client	13
4.2.2	RedBack::Links	13
4.2.3	RedBack::CCI	13
4.2.4	RedBack::Model	14
4.2.5	RedBack::Graph	15
4.2.6	Bugs in Perl/Tk	16
4.2.7	Source code	16
5	Issues in assisting WWW navigation	17
5.1	Programming	17
5.2	User interface	18
5.3	World Wide Web	18
5.3.1	HTML problems	18
5.3.2	URL problems	18
6	Further work	20
6.1	Studies of the existing version	20
6.2	Extending the program	20
6.3	Other uses	21
7	Summary	23

Chapter 1

Introduction

The World Wide Web (WWW) has become a very popular tool in recent years for providing information of all kinds. However, because of the ease with which this information can be linked together in a loosely structured way, users may have difficulty finding information and keeping track of the locations of pages they have looked at recently. These problems with hypertext and the WWW have been known for several years [15]. This is generally known as being “lost in hyperspace.”

This project involved the investigation of a part of the problems associated with navigation of the World Wide Web. A program was developed to investigate one approach. This program was given the name **RedBack**, since the idea of a spider is related to the World Wide *Web*, and the ‘Back’ button in common browsers is part of much WWW navigation.

Some of the goals of this project included the following:

Review related work

The issue of WWW navigation has been approached by other researchers. Their research should provide valuable input to this project.

Develop a working prototype

The program developed to investigate the problems should have sufficient functionality and be robust enough to allow it to be properly evaluated.

Provide for future work

The prototype should not be limited to its particular application, but should be understandable and extensible to related projects.

Investigate future work

A one year project is limited in scope, so consideration must be given to ways in which it could be extended.

The remainder of this report presents the work which was done toward these goals. First, chapter 2 describes some previous work in related fields. The following chapter gives a thorough description of the development and implementation of the **RedBack** program, and chapter 5 then discusses some of the issues raised during program development.

Chapter 6 then describes further work that would be possible on the RedBack program, and the last chapter summarises this report, and the results of the project.

Chapter 2

Previous work

This project involves fields of computer science which build upon earlier and more general work. At the most general level, it involves *hypertext* systems. Topics more specific to the *World Wide Web* and navigation in particular are based on this work. This project also has a more specific focus on visualisation of hypertext and WWW systems, where applicable to the problems of navigation.

2.1 Hypertext and World Wide Web

Research into the field of hypertext goes back many years. Some research relevant to this field even predates computers [6], although the term “hypertext” itself was only coined by Nelson around 1965 [14].

The World Wide Web is based upon hypertext, but its use of the Internet allows access to any Internet-connected host from any other. The technology was originally developed as a way for scientists at CERN to distribute information quickly and easily, but it has become applicable world wide [2]. This is partly coordinated by the Internet Engineering Task Force (IETF) which provides standards for the content (HTML [3]) and transfer protocols (HTTP [4]).

A number of code libraries have been developed and are freely available to allow access to the WWW as defined by these standards without the programmer needing to ‘reinvent the wheel.’ An example is the WWW Library¹ from the World-Wide Web Consortium (W3C).

2.2 Navigation

The issue of how people access the World Wide Web, from the viewpoint of how they navigate from one document to another as they look for information, has been investigated by several researchers. Their results reveal deficiencies of current browsers which influenced the design of the RedBack program.

Catledge and Pitkow’s research [7] focussed on the use of a modified version of NCSA Mosaic to record the actual use of a web browser. This data was analysed to extract patterns and determine the techniques used while browsing the WWW.

¹<http://www.w3.org/pub/WWW/Library/>

The findings with most relevance to this project were the relative frequencies of different methods of access to documents. The most preferred method was by selecting hyperlinks on pages, which accounted for approximately 52% of the document requests collected, then the use of the 'Back' button accounted for another 41%. All other methods such as the hotlist, history or the 'Forward' button each accounted for 2% or less.

Another relevant finding was that users would follow a 'hub-and-spoke' approach to browsing, i.e. they would start from one page, the hub, and visit pages two to three links away before returning to the hub using either the 'Back' button or a link back to the hub on the visited page.

The usability problems associated with WWW navigation were investigated by Cockburn and Jones [8] in a small usability study. This investigated the use of the history mechanisms provided by the Forward and Back buttons in conjunction with links back to the hub as described above, and the accuracy of the users' models of the behaviour of these buttons.

The behaviour of these buttons in the Netscape Navigator and NCSA Mosaic browsers is such that simple actions can remove recently visited pages from the history. Using the terminology of a hub-and-spoke system, after a user has navigated to a page along a spoke and returned to the hub using the Back button, the pages along the spoke are still available using the history list or the forward button. However, after selecting another link from the hub, the previous spoke is removed, and the pages on it are now inaccessible except by selecting links which lead to it again.

In their study, Cockburn and Jones discovered that many of the subjects had incorrect mental models, expecting all previously visited pages to be available.

Recent work by Tauscher and Greenberg [17] has further investigated the use of history in WWW browsing. They used their own modified version of NCSA Mosaic to collect data in a similar manner to Catledge and Pitkow [7]. Analysing their own data in conjunction with interviews of their subjects, and combining this with Catledge and Pitkow's original data, provided more accurate insights into how people use the WWW.

Tauscher and Greenberg identified seven different browsing patterns. Their analysis also included the frequency with which URLs were accessed, and the frequency as a function of the distance, i.e. the number of pages seen since a revisited page was last seen. In the latter case they found that there was a fairly high probability that a revisited page would have been seen recently. The previous six pages visited contain the majority of pages likely to be visited next.

Using this information, a number of different history mechanisms were proposed. These are described below.

Recency ordered

In this kind of history, pages visited are sorted by time, with most recent at the top of the list. Variations on this method are to eliminate duplicates, by either removing the older entry, or not inserting the newer one.

Frequency ordered

A frequency ordered history list has the page that has been visited the most times at the top of the list.

Stack based

This is the method used by NCSA Mosaic and Netscape Navigator which

Cockburn and Jones investigated [8].

Hierarchically structured

The two methods employing hierarchical structuring that were examined by Tauscher and Greenberg were *recency ordered hyperlink sublists* and *context-sensitive Web subspace* lists. The first of these is similar to the recency ordered history except that from each page, pages that were visited from that page can be accessed from a second list. The second type of hierarchically structured list is based upon a design put forward by Cockburn and Jones [8], where pages are organised into ‘subspaces’ identified by the first page ‘directly accessed’.

These proposed mechanisms were then evaluated according to the previously collected data. The authors acknowledged that this did not account for the effect the model had on the users’ browsing patterns.

2.3 Visualisation

A number of programs have tried to solve the problems associated with WWW navigation using various kinds of graphical views. Other research is also relevant to this area.

An early program to provide a graphical overview was WebMap [9]. By making modifications to NCSA Mosaic version 2.5, the WebMap program was able to read each HTML page retrieved by Mosaic. It used this information to present a record of pages visited as a tree, where each node represents a page and is numbered according to the order in which it was visited.

The paper does not seem to be very clear on how the pages are arranged given the links on the page, and a tree view does not represent how the WWW is actually arranged, or explored. In addition, the display of a single number for each page does not provide much feedback on the identity or content of each page in the overview.

The Navigational View Builder [12] looks at a set of WWW pages on one server and provides a number of different ways to display two and three dimensional views of the pages, and links between them.

The main method uses a two dimensional display of an undirected graph, where links between pages are represented by a straight line between the squares or circles representing the pages. The graph of all pages on the site can be very large in size, so a number of techniques are used to add information, or reduce the number of pages displayed.

Some of the methods presented in the paper to improve the display were:

Binding

Icons, shapes, colours and patterns could be added to pages and links depending on various properties such as type of page (HTML, plain text or a picture for example), file size, author and topic.

Clustering

Pages could be grouped together into a single node according to various criteria, such as having the same author or similar topics. This could be combined with binding to identify each group of pages.

Filtering

Using similar criteria to those used for clustering, specific subsets of pages can be shown on their own.

The Navigational View Builder could also use different kinds of displays to sort the pages hierarchically. Hierarchies could be formed on a number of criteria, in much the same way as the clustering and filtering.

All these features make the Navigational View Builder a powerful tool for displaying an overview of a WWW site. However because the view is statically generated, changes in pages are not reflected immediately. It also appears that the Navigational View Builder is only useful for displaying the overview separately, as no mention is made of it having an interface with any WWW browser.

An approach which was aimed at enhancing the history mechanism is taken by MosaicG [1]. It is integrated into the source code of NCSA Mosaic itself, which allowed it to provide a few more features.

The history presented by MosaicG takes the form of a tree built from left to right. Each page that is visited is made a descendent of the page that it was selected from, unless it already exists. Although this does not describe the structure of the World Wide Web very well, it is consistent with some of the browsing patterns that have been observed.

To provide a visual cue to the user about which page is which, MosaicG displays a reduced snapshot of the top left corner of the page after Mosaic has loaded it. It also includes an abbreviated version of the page's title.

MosaicG also provided mechanisms to zoom in or out of the page, increasing or decreasing the amount of detail shown for each page, and respectively decreasing or increasing the number of pages that would fit on the screen at once. It was also possible to collapse all the pages below one node of the tree into the space of the top node.

However, Ayers and Stasko did note that in preliminary reviews of the program, some users would have liked more control over the structure of the tree. Features such as erasing sub-trees completely or moving a sub-tree to the root of the tree were proposed.

The Merz project [11] also provides a two dimensional graphical overview of WWW pages. The emphasis in this case is on allowing the user to organise the pages they have seen, so the software lets them rearrange the objects representing the pages within the overview, and in separate collections.

The pages are not automatically labelled, but the users can add notes or labels themselves. The Merz software also allows to add their own links between related pages. Methods for selecting or filtering pages like the Navigational View Builder [12] are provided.

Other aims of the Merz project include allowing collaboration by groups of users on the collections put together by individuals, and the use of sound and three dimensional views is also the subject of research.

Chapter 3

RedBack design

Decisions made during the design process can be separated into a few different categories. These are described in this chapter.

3.1 Program scope

RedBack could provide a wide range of different features, to the extent of incorporating most of the features of all the programs mentioned in section 2.3. To get this working would be quite difficult and could take a lot of time, and the result might not show anything useful anyway.

For these reasons RedBack represents just a first step towards a WWW visualisation tool that could run on a desktop computer. A simple two dimensional graphical overview is used to present information in a directed graph, similar to the Navigational View Builder [12] or other programs.

To simplify the management of information and the display, RedBack is intended mainly to present a solution to the problems with keeping track of the user's location in hyperspace and relationship to other pages seen recently, similar to the aim of Mosaic G [1]. Among other things, this means that it is not necessary at first to implement the features described in chapter 6 on further work, although including all the features possible would be desirable.

3.2 Code organisation

The decision was made to use an object-oriented paradigm in the development of RedBack. This was influenced by the provision in Perl for object orientation (see section 4.1.2 below), but was mainly due to the author's feeling that this approach is easier to program with and produces more understandable and extendible code. The last point is especially important because RedBack is intended as a simple program that can lead to a more complicated and more powerful implementation.

As well as the object oriented approach, much of the communication between objects was achieved with *call-backs*. One instance of an object requests notification of events from another by passing it a subroutine to be *called back* to whenever such events happen.

The functionality of the program was separated according to purpose and data used, to provide specifications for interfaces to abstract objects. Descriptions of these objects are given below.

Interface supplies methods for accessing any WWW browser. This could be applied to communication with a browser that is part of the same program, or over some protocol like using CCI with Mosaic. The parts of the program using the interface should not need to know what browser they are talking to. In keeping with the object oriented approach, it should be possible to have many instances of any number of classes based on this abstract class.

Model stores all the data associated with the program's current knowledge of the real WWW, as far as is necessary. Various methods provide ways for the interface to add an entire HTML page and for the display below to selectively request information like links leading to and from a page or the best text so far for the description of a page. Ideally other objects should not need any knowledge of the way the data is stored.

Display presents the data based on what is stored in the model, and what pages have been displayed by the browser.

3.3 Graphical display

The design for the graphical display provided by RedBack is based in part upon the programs described in section 2.3. It presents the display as a directed graph like the World Wide Web itself is organised.

Three different features should be displayed by RedBack:

1. Visited pages, which should be displayed as nodes of the graph. Some sort of description or title should be included, although to save space it will need to be abbreviated in some way.
2. Links between the pages described above. They should use some kind of indication such as arrowheads to indicate which page the link is on, and which is the destination.
3. There should also be some indication of unvisited links from each page, although this could be difficult on pages with hundreds of links. In this case it would probably be necessary to visit the page anyway.

As the mouse moved over each page, link between pages or unvisited link, it could be highlighted as a visual cue as to what is selected. For pages and unvisited links, a description of the page and its URL should be displayed somewhere in the window.

For RedBack's communication back to the browser, clicking on any visited page, or on a link to an unvisited page, should fetch and display that page.

When a new page is displayed as a result of selecting it in RedBack or in the browser, RedBack's display should be updated by adding the new page and links to it where appropriate.

Chapter 4

RedBack implementation

4.1 Implementation environment

This section covers the decisions made as to which programs and languages were used in the development of RedBack. The impact of these decisions is discussed in the following chapter. It also includes a description of the code of RedBack itself.

RedBack was developed using NCSA Mosaic as the browser, and Perl as the language the program was written in. Perl/Tk was used for the user interface. The following sections describe these, and why they were chosen.

The X Window System (X11) and UNIX were the underlying environment for RedBack. Part of the reason for this is that they were the immediately available for development, but they also offer other advantages such as a good development environment and platform independence.

4.1.1 NCSA Mosaic

If RedBack is to be useful to a normal user, it would help to present a familiar interface at least partly. The most popular WWW browser for X11 is Netscape Navigator. However NCSA Mosaic has a similar interface which should be familiar to anyone who uses Netscape, and it has other advantages besides the fact that it is free to use. An example display of NCSA Mosaic is shown in figure 4.1.

Versions of Mosaic since 2.6 have included a feature called the Common Client Interface (CCI) [13]. This is a form of inter-process communication which lets other programs communicate with Mosaic. In particular, the following features are useful:

- CCI uses standard TCP/IP for communication. This makes it possible to use the interface to communicate with Mosaic from any language that provides access to sockets. There is an API available from NCSA that provides a standard method of accessing CCI from C and from older versions of Tcl and Perl. Its use is not compulsory though, because the actual protocol is fairly simple.

The biggest problem with this method of communication is that it is difficult for the client to find out where to connect to. For clients running from

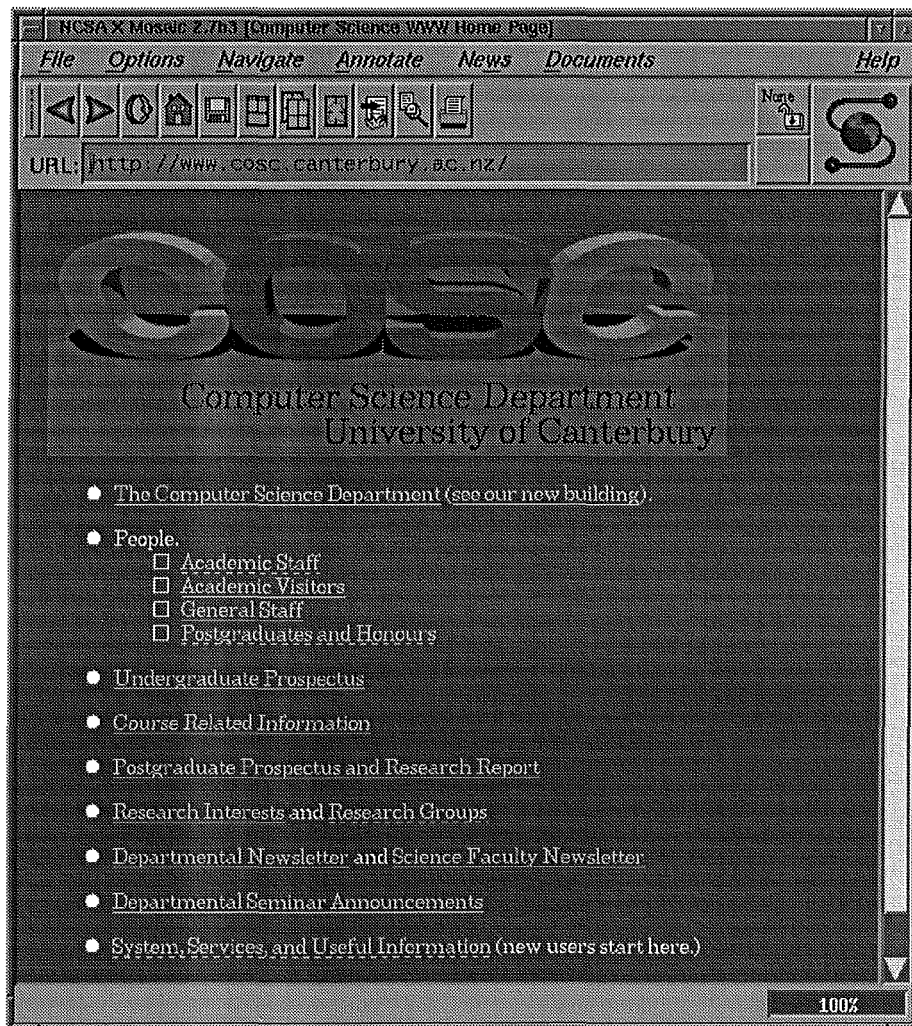


Figure 4.1: NCSA Mosaic

the same account as the browser, the file `~/mosaiccciport` contains the name of the host and the port number to connect to. Netscape Navigator uses X Window System properties to let any application on the same display send it commands [19], but it does not provide any mechanism to retrieving the contents of the page retrieved.

- With the `SEND BROWSERVIEW` command, Mosaic will send back the raw HTML code of every WWW page that is displayed, as well as the location of that page as a URL, to the CCI client. This lets RedBack extract all the information it wants about the page and the links from the page.

This does lead to some duplication of effort as unlike MosaicG [1], where all the results of Mosaic's parsing are available, each document must be parsed by both Mosaic and the client.

- Based on the data received by the previous method, or on other user input, the client can instruct Mosaic to retrieve and display any page given by its URL. Like all other pages viewed, the content of the page will be returned to the client if requested using the method above.

Some problems exist with the CCI method. As mentioned above, there is some duplication of effort in extracting the links. Mosaic also provides no information about how each page was retrieved, such as by clicking on a link, submitting a form, as a redirection from another address or by any of the other methods investigated by Tauscher and Greenberg [17] or Catledge and Pitkow [7]. The right support in CCI might have made the modifications to Mosaic by these researchers unnecessary and could also supply extra useful information to RedBack.

Other approaches would have been to modify a copy of Mosaic to integrate RedBack with the browser's source code or to send processed information to RedBack, or to modify one of the browsers written in Tcl/Tk [16] or Perl/Tk, but the separation makes it easier to debug RedBack without looking at the Mosaic source code. Another approach which was not considered at the beginning of the project would be to implement RedBack as a Java applet running in Netscape Navigator or as part of Sun's HotJava browser, which would offer it full access to the browser's information and actions. The increasing popularity of Java over the year of 1996 has made this approach seem more realistic.

4.1.2 Perl

Although a predecessor of RedBack was written in Tcl, the decision was made to use Perl [18] as the language in which to implement the program for this project. This section explains why Perl itself was chosen; the following section describes the `libwww-perl` modules for working with the WWW from Perl.

Tcl/Tk is often used for producing prototypes of programs that have a graphical user interface. There are several reasons for this, including the following:

- The Tk user interface toolkit makes it quite easy to quickly build a simple user interface.
- Because it is interpreted, any changes in Tcl code can be immediately reflected in the program without recompilation.

However it also has some problems derived from the initial design of Tcl being a simple scripting or extension language, so that it is completely interpreted, even each time through a loop, and it has no support for complex data structures. The only data structures in Tcl are strings and associative arrays. Everything in Tcl also has the same name-space, so that all variable and procedure names in the whole program must be unique.

Perl has both the advantages of Tcl listed above, but does not suffer from the disadvantages. Perl has the following features:

- After the program code has loaded, it is ‘compiled’ to an intermediate form which can be executed a lot faster.
- Built in data types are strings, lists, ‘normal’ arrays and associative arrays. The strings can contain arbitrary binary data, unlike Tcl. By using object orientation, other data types can be defined.
- Code can be organised into separate modules, each with its own name-space that does not interfere with other modules.

Perl/Tk was used for the user interface for RedBack. It is a port of the Tk part of Tcl/Tk to Perl. This provides a familiar and relatively easy to use interface to RedBack, which is also fairly easy to program. The current version of Perl/Tk has reached the stage where it is no longer considered to be a beta test version.

4.1.3 libwww-perl

Libwww-perl is a large collection of modules for Perl which provides methods for working with the World Wide Web, like the W3C WWW library for C. These routines provide tested methods for accessing and using WWW related data.

The following modules, actually object oriented classes, are used by RedBack.

URI::URL

This module provides manipulation of Uniform Resource Locators (URLs) [5] and resolution of relative URLs [10] within HTML documents. The main use of this module within RedBack is to convert URLs to a standard form, after stripping off the fragment description (i.e. #fragment at the end of a URL).

This was used so that, for example, the equivalent URLs in table 4.1 would all be resolved as the same page.

<pre>http://www.cosc.canterbury.ac.nz/~carey/links/rfc-u.html http://www.cosc.canterbury.ac.nz/%7Ecarey/links/rfc-u.html http://www.cosc.canterbury.ac.nz:80/~carey/links/rfc-u.html http://www.cosc.canterbury.ac.nz/~carey/links/rfc-u.html#mime</pre>
--

Table 4.1: Equivalent URLs

HTML::Parser

Two concrete implementations of parsers derived from this abstract object are used by RedBack. The `HTML::HeadParser` parses just the `<HEAD>` section of an HTML document. RedBack uses the `<TITLE>` and `<BASE>` of the HTML header.

The `HTML::TreeBuilder` class converts the entire HTML document into a syntax tree representing the document. RedBack then traverses this tree extracting links, and getting the text of each link to possibly be used as a description.

4.2 The RedBack program

This section contains a description of the implementation of the RedBack program.

4.2.1 CCI::Client

Mosaic CCI APIs are available from NCSA for C, Tcl and Perl 4 [13]. To allow RedBack to be implemented in Perl 5, a separate module was written encapsulating all the necessary behaviour. Each connection is implemented as an instance of an object, and communication with Mosaic is set up by the object's constructor.

This module provides an interface like that of many other Perl libraries to access Mosaic. It doesn't supply methods for accessing all the functionality of CCI, but the behaviour needed by RedBack is present.

4.2.2 RedBack::Links

For testing the `CCI::Client` module, and later the `RedBack::Interface` `RedBack::Model` modules, a small program to list the links on the current page was developed and later encapsulated into its own `Display` class. For the WWW page shown in figure 4.1, the Links display would be that shown in figure 4.2.

Although it was not part of the original design, this program is quite useful on its own for quickly selecting one of a number of links from a page. Two additional features were added to the original. The "Lock page" button toggles between updating the list when Mosaic retrieves a new page, or leaving the same links visible. The "New links" button creates a duplicate containing the same page. Locking on one page can be quite useful for hub-and-spoke browsing, so that other spokes can be selected without using 'Back' to return to the hub.

4.2.3 RedBack::CCI

This module provides an abstracted interface to the `CCI::Client` module, hiding the details of what is being connected to. It provides methods to:

Connect by creating a new instance of the interface.

Disconnect from the browser by closing down the communications.

Fetch a page by sending the appropriate command to Mosaic.

Add a callback and remove it to find out when a new page has been retrieved or the connection has been closed.

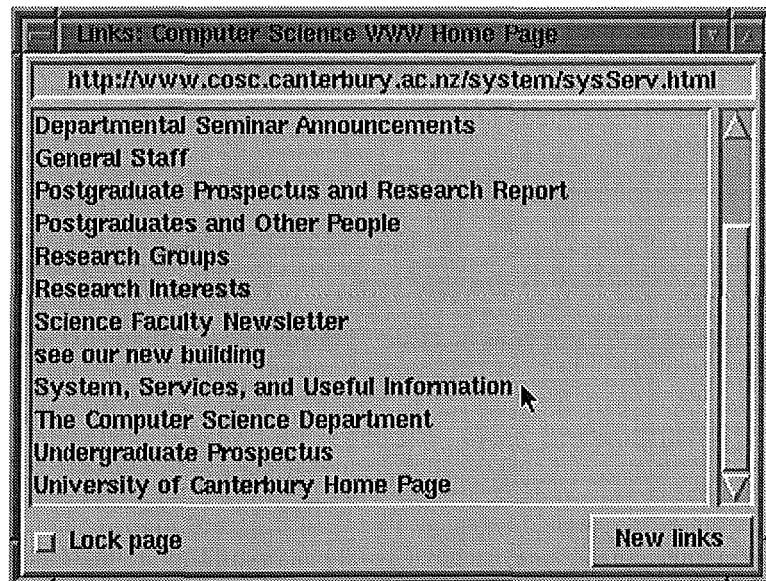


Figure 4.2: RedBack 'Links' display

4.2.4 RedBack::Model

This module stores all the information that is used by RedBack about the pages that have been seen. The following information about every URL seen is currently stored:

Type of URL

Whether the URL has been seen just as a link on a visited page, or whether it has actually been visited.

Links on the page

All the URLs of links on the page are recorded, if applicable.

Links from other pages

The URLs of pages which contain links to this page are also included.

MD5 checksums

The MD5 checksum is calculated for each page seen. This is currently only used to decide whether to rescan for links if the page has changed since it was last visited. However it could also be used in conjunction with the MD5SRC attribute on <A> tags to check whether the page appears to have been tampered with.

Description

A description is derived from whatever is known about the document. Different sources are given different priorities as described below, but otherwise conflicting descriptions are decided in favour of the first one seen.

Time

This is the date and time that the page was last seen by RedBack. It is not currently used for anything.

Descriptions for pages are ranked according to the following sources:

1. <TITLE> tag of the document itself.
2. TITLE attribute of an <A> tag.
3. Text between the <A> and tags, including ALT attributes on images.
4. The URL of the page, as a last resort.

Some methods are provided for accessing this data. Unfortunately not enough planning went into this, and procedures requesting the above information generally have to extract it themselves.

4.2.5 RedBack::Graph

This module provides the display that was the purpose of this project. After browsing a few pages, a display like that in figure 4.3 is obtained.

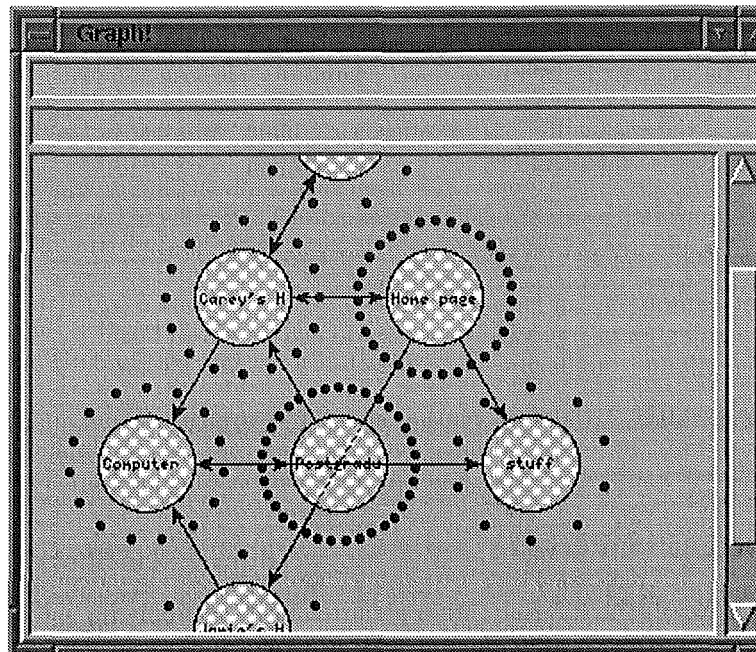


Figure 4.3: RedBack graphical display

The display is much like that described in the design in section 3.3. Each of the visited pages is displayed as a hollow circle with the first nine letters of the description described above in the centre. The links are shown as straight lines

with arrowheads joining the pages. Unvisited pages are shown as the black dots ‘orbiting’ the visited pages.

The appearance of the pages, including orbiting links, and links between pages are controlled respectively by the `RedBack::Graph::Node` and `RedBack::Graph::Link` classes. When an instance of one of these is created, it draws itself on the canvas, and it removes itself from the canvas when the Perl garbage collector removes it after all references to it have been discarded.

When the mouse moves over parts of the display they are highlighted. In all cases the colour is changed to red. For visited pages, the thickness of the border is increased. For links between pages, the thickness of the link is increased and the pages at each end are also highlighted. For unvisited pages, the radius of the dot is increased, for *all* dots linking to the same page.

Pages are added to the graph as they are displayed by Mosaic after the graph is displayed. The `RedBack::Graph` class decides where new pages will be placed. The algorithm it uses, which is not based on any existing research, is summarised below. Pages can only be positioned on an isometric (triangular) grid.

1. If there are no existing pages on the display, just put the new one in the middle. Otherwise, carry out the remaining steps.
2. Work out the positions of all the pages with links to or from the new one, or all the pages if there are no links.
3. Search outwards from these positions to find all the gaps at a certain number of steps from any of these pages.
4. Choose the gap with the lowest average distance from all the positions and put the new page there.

This seems to work reasonably well for keeping all interconnected pages together, at least while the number of pages is small.

There are some problems with this display:

- If the number of unvisited pages is too high, they all blend together into a ring around the page.
- The links overlap with each other and with the pages.
- Even a few pages takes up a lot of screen space.

4.2.6 Bugs in Perl/Tk

Unfortunately this program has proven to be unstable, and will crash after a period of use. The most common type of crash appears to be a bus error in the Perl code. Since the Perl language provides very little direct access to memory, none of which is used by `RedBack`, it seems that the cause of the crash must be a bug in the C code of either Perl or Perl/Tk.

4.2.7 Source code

The full source code for `RedBack` is available from the author.

Chapter 5

Issues in assisting WWW navigation

Implementing RedBack and using the prototype version provided insight into several issues associated with programming, user interface and World Wide Web use.

5.1 Programming

At the beginning of the project, Perl and its associated libraries were selected as the environment to use in the development of RedBack. This had advantages and disadvantages.

The object oriented approach enabled by Perl worked well to separate out functionality into different modules. This also helped development with the Model and Interface being worked on with the “Links” display, but with the graphical display slotting right in when it was finished.

However, this approach was not extended to the values returned by the Model, resulting in code that was very dependent on the internal format in which data was stored, and was also difficult to understand.

Another place where the object oriented approach could have been applied better was in a better separation of duties within the graphical overview. Policies for displaying the graph and maintaining information about it could be abstracted into separate classes allowing easier development of alternative displays and user interfaces.

The libwww-perl World Wide Web libraries available for Perl made extracting information like the URLs and text content of links relatively simple, and the URL::URL module especially removed a lot of work in comparing URLs. Unfortunately it runs noticeably slower than Mosaic in parsing the page and extracting links.

Implementing an interface to the Mosaic Common Client Interface was not especially difficult, given the simplicity of the protocol. Some problems were encountered with the conflict between needing to buffer the incoming data, but to still make sure that there was any new data would show the socket as readable to the calling program, but this was resolved without too much trouble.

5.2 User interface

When a user selects a page or link from the RedBack display, the command to fetch that page is sent to Mosaic and RedBack returns to its previous state. It cannot wait, because if the user interrupts the retrieval it will never finish waiting. In development versions, this meant that the only feedback the user received was of Mosaic itself loading the page.

This might not be immediately obvious, and could lead to the user repeatedly clicking until the Mosaic display actually updated. For this reason whenever a user clicks on a page or link, the mouse cursor is changed to the watch symbol for half a second, to give him the impression that something is happening even if he is not keeping an eye on Mosaic.

5.3 World Wide Web

Two major problems were discovered with RedBack that were related to the World Wide Web. The first is related to HTML, and the second to how URLs work.

5.3.1 HTML problems

Partly through the leniency of programs such as Netscape Navigator in interpreting HTML, there are a lot of pages in existence that do not conform to official standards such as that for HTML 2.0 [3]. The libwww-perl libraries accepts some variations but not all. For example, the HTML::HeadParser will not retrieve any information from a header like that in figure 5.1, so the page title will not be extracted.

There are also some cases where libwww-perl will not extract the text of an `<A>` tag because it is used in a non-standard way.

```
<HTMLPLUS>
<HEAD>
<TITLE>Document using old, obsolete HTML+ specification</TITLE>
</HEAD>
```

Figure 5.1: Incorrect HTML header

5.3.2 URL problems

The second problem is that there are many different ways to retrieve the same page using different URLs [5]. For example, the URLs in tables 5.1, 5.2 and 5.3 all retrieve the same page¹, although the URLs within each table are all distinct addresses.

¹Some machines listed are only accessible from within the University of Canterbury.

<code>http://www.cosc.canterbury.ac.nz/</code>
<code>http://www.cosc.canterbury.ac.nz</code>

Table 5.1: URLs with different trailing slashes

<code>http://www.cosc.canterbury.ac.nz/</code>
<code>http://cosc.canterbury.ac.nz/</code>
<code>http://huia/</code>

Table 5.2: URLs on different hosts

The problem demonstrated in table 5.1 occurs very often when referring to folders. The HTTP server usually redirects the browser to the URL with the trailing slash, which is the correct notation for a folder. However, any links to the URL without the trailing slash will always seem to be unvisited.

In table 5.2, there are two problems: `cosc.canterbury.ac.nz` and `huia` both resolve to the machine at the IP address 132.181.10.11, and both this machine and `www.cosc.canterbury.ac.nz` (132.181.10.25) are running HTTP servers supplying the same pages.

Even the former case has no simple solution, since the same machine can supply different pages depending on the URL given. In the URLs `http://archie.nz/` and `http://sunsite.net.nz/`, the same host (140.200.128.20) supplies the page, but different pages are supplied for each URL.

Finally in table 5.3 there are also two causes of the problem. The first two URLs retrieve the same page because the HTTP server in this case returns the contents of the file `index.html` whenever a request is received for the folder itself, and in the second case because this site has a symbolic link from `index.html` to the original location in `Homepage.html`.

A related problem can occur on some hosts, mostly those using operating systems from Microsoft that support short, all capital filenames. In these cases, URLs such as `http://www.psyc.canterbury.ac.nz/intro.htm` and `http://www.psyc.canterbury.ac.nz/Intro.html` will fetch the same page.

<code>http://www.cosc.canterbury.ac.nz/</code>
<code>http://www.cosc.canterbury.ac.nz/index.html</code>
<code>http://www.cosc.canterbury.ac.nz/Homepage.html</code>

Table 5.3: URLs addressing copies of pages

Chapter 6

Further work

An important step before any further work is carried out would be to resolve the problem that causes frequent crashes of Perl when **RedBack** is run. The authors of Perl and Perl/Tk seems receptive to new ideas and bug reports, so if a bug is found it should be fairly easy to get it fixed in a new version. On the other hand, it may be possible to change the program code to work around the code which causes the crash.

Once a stable version of **RedBack** is available, the ideas below could be investigated.

6.1 Studies of the existing version

It would be very useful to do a complete usability study of the different features of **RedBack**, like that done for **MosaicG** [1] from which a number of interesting ideas were obtained.

At this stage, many of the ideas for different features and evaluation of the program has been only the author's, which cannot lead to a good program due to the author's knowledge of exactly how the system behaves.

Usability studies would probably provide even more ideas for new features than just those given below.

6.2 Extending the program

RedBack was intentionally implemented as a very simple system. Given its modular design, there is some scope for extending the program to investigate other views or uses of WWW browsing history.

Graphical overview enhancement

MosaicG, the **Navigational View Builder** and the **Merz** project all implement features in their graphical overview that **RedBack** does not [1, 11, 12]. Some of these are described below.

Better algorithm

The current algorithm described in section 4.2.5 does not perform very

well. Better mechanisms for initial placement of pages, and for choosing the path of the links, could have a significant impact on the usability of the graphical overview.

Direct manipulation

It would be very useful to be able to rearrange the positions of pages on the graph by some kind of direct manipulation with the mouse. Other changes could be allowed such as removing a page entirely, or temporarily hiding or collapsing sets of pages.

Filters, highlighting and clustering

As the graph shown grows, it would be useful to be able to restrict the view to just part of it by filtering based on keywords or sites. Highlighting could be based on the same criteria, and they could be applied to grouping related pages together in the graph.

The filters and highlights need not affect just pages on the graph, but could also be used to restrict the visited and unvisited links displayed.

Annotations

As well as letting users arrange pages as they wish, they could also add extra information by annotating the pages. This could be shared with NCSA Mosaic's annotations system.

Enhanced history lists

As noted by Tauscher and Greenberg [17], a graphical overview is not the only way to improve the recall of pages. The 'Links' part of RedBack could be modified to provide a number of the different history mechanisms, and this could be used to study the effect of having these mechanisms available, rather than just applying them to observed browsing habits.

Retaining data over sessions

Currently each graph is only updated with pages visited after that graph was created. This could be improved in two ways, given the filtering and highlighting described above so that pages do not accumulate too much.

First of all, all the pages visited should be accessible from another graph created during the same session, if the user chooses to do this. It should also be possible to specify that only pages that were visited during a certain period are included.

There should also be provision for saving data about the visited pages before exiting RedBack and reloading it next time RedBack is run. Despite the complex data structures used by the Model, Perl libraries such as the Data::Dumper should actually make this fairly simple.

6.3 Other uses

A graphical overview like RedBack need not only be used as a sort of long term history or 'hotlist'. Some other uses are described below.

Authoring

As noted in section 5.3.2, there are a number of cases in which different URLs can mean the same thing. The problems are even greater when the URL on a page does not link to anything at all, because it has been entered incorrectly.

RedBack could have applications in checking sets of pages for broken links. With some extensions it could provide features for adding links, for example by dragging a new link from one page to another on the graph, and automatically adding a link to the starting document's HTML.

Generating maps

With a better graph layout algorithm, RedBack could be used much like the Navigational View Builder [12] and some other programs for generating maps of a WWW site.

Chapter 7

Summary

This project has identified a number of issues related to problems in assisting navigation while browsing the World Wide Web, and of getting “lost in hyperspace.”

Previous research has identified some of the problems people have with existing browsers and the way in which people use these to browse the WWW. Other work has gone into developing programs which attempt to enhance the use of the WWW by providing graphical overviews.

A program, **RedBack**, was developed that provides a graphical overview of previously visited WWW pages. Although it is currently unstable, the separation of code into separate modules should make it a good base for investigation of different issues in WWW navigation. The development and use of **RedBack** has highlighted some problems with navigation and the design of parts of the World Wide Web.

Bibliography

- [1] AYERS, E. Z., AND STASKO, J. T. Using graphic history in browsing the world wide web. In *Fourth International World Wide Web Conference Proceedings* (Boston, MA, US, Dec. 11–14 1995).
- [2] BERNERS-LEE, T., CAILLIAU, R., LUOTONEN, A., NIELSEN, H. F., AND SECRET, A. The world-wide web. *Communications of the ACM* 37, 8 (1994), 76–82.
- [3] BERNERS-LEE, T., AND CONNOLLY, D. W. Hypertext markup language — 2.0. Internet RFC 1866, Nov. 1995.
- [4] BERNERS-LEE, T., FIELDING, R. T., AND NIELSEN, H. F. Hypertext transfer protocol — HTTP/1.0. Internet RFC 1946, May 1996.
- [5] BERNERS-LEE, T., MASINTER, L., AND MCCAHILL, M. Uniform resource locators (URL). Internet RFC 1738, Dec. 1995.
- [6] BUSH, V. As we may think. *The Atlantic Monthly* 176, 1 (June 1945), 101–108.
- [7] CATLEDGE, L. D., AND PITKOW, J. E. Characterizing browsing strategies in the world wide web. In *Computer Networks and ISDN Systems: Proceedings of the Third International World Wide Web Conference. Darmstadt, Germany* (Apr. 10–14 1995), vol. 27, pp. 1065–1073.
- [8] COCKBURN, A., AND JONES, S. Which way now? analysing and easing inadequacies in WWW navigation. *International Journal of Human-Computer Studies* 45, 1 (1996), 105–129.
- [9] DÖMEL, P. WebMap — a graphical hypertext navigation tool. In *Second International World Wide Web Conference Proceedings* (1994).
- [10] FIELDING, R. T. Relative uniform resource locators. Internet RFC 1808, June 1995.
- [11] LENMAN, S., AND SEE, H. Personal information spaces. In *Poster Proceedings of the Fourth International World Wide Web Conference* (1995), pp. 56–57.
- [12] MUKHERJEA, S., AND FOLEY, J. D. Visualizing the world wide web with the navigational view builder. In *Computer Networks and ISDN Systems: Proceedings of the Third International World Wide Web Conference. Darmstadt, Germany* (Apr. 10–14 1995), vol. 27, pp. 1075–1087.

- [13] NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS. NCSA Mosaic common client interface. Available from: <URL:<http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/CCI/cci-spec.html>>, Mar. 31 1995.
- [14] NELSON, T. H. *Dream Machines: New Freedoms Through Computer Screens—A Minority Report*. Self-published, 1978. Issued with “Computer Lib”.
- [15] NIELSEN, J. The art of navigating through hypertext. *Communications of the ACM* 33, 3 (1990), 296–310.
- [16] OUSTERHOUT, J. *An Introduction to Tcl and Tk*. Addison-Wesley, 1993.
- [17] TAUSCHER, L., AND GREENBERG, S. Revisitation patterns in world wide web navigation. Research report 96/587/07, Department of Computer Science, University of Calgary, Canada. Available from <URL:<http://www.cpsc.ucalgary.ca/projects/grouplab/papers/96WebReuse/TechReport.html>>, Mar. 1996.
- [18] WALL, L., CHRISTIANSEN, T., AND SCHWARTZ, R. L. *Programming Perl*, 2 ed. O’Reilly & Associates, Inc., Sept. 1996.
- [19] ZAWINSKI, J. Remote control of Unix Netscape. Available from <URL:<http://home.netscape.com/newsref/std/x-remote-protocol.html>>, 1996.

Notes on bibliography

Internet Requests for Comments are available from many locations, including on the WWW at <URL:<http://ds.internic.net/ds/rfc-index.html>>.